Student Name: Guangbo Chen
Student ID: 11376860

Advanced Internet Programming 31242

– Assessment 2 report

Subject Coordinator: Ryan Heise

- 2013 Spring Semester

Group Member: Guangbo Chen

Student ID: 11376860

Task Distribution:

Tasks	JPA	EJB	Web Application	Web Service	Database	Report
Group Member	Guangbo Chen	Guangbo Chen	Guangbo Chen	Guangbo Chen	Guangbo Chen	Guangbo Chen

System Overall Architecture

The architecture that we have implemented in our project is N-tiers architecture; this is typical architecture for enterprise web application with highly scalability and build from reusable components, it also supports distributed applications.

Database: In this project we have employed MySQL as our database, it is easy to install and maintain.

JPA: The Java Persistent API that we have used in this project is Hibernate; Hibernate is the most up-to-date technology that most users used for their applications, Hibernate facilitated the storage and retrieval of Java domain objects via Object/Relational Mapping, it is open-source and free to use with plenty of documentations supported by the provider.

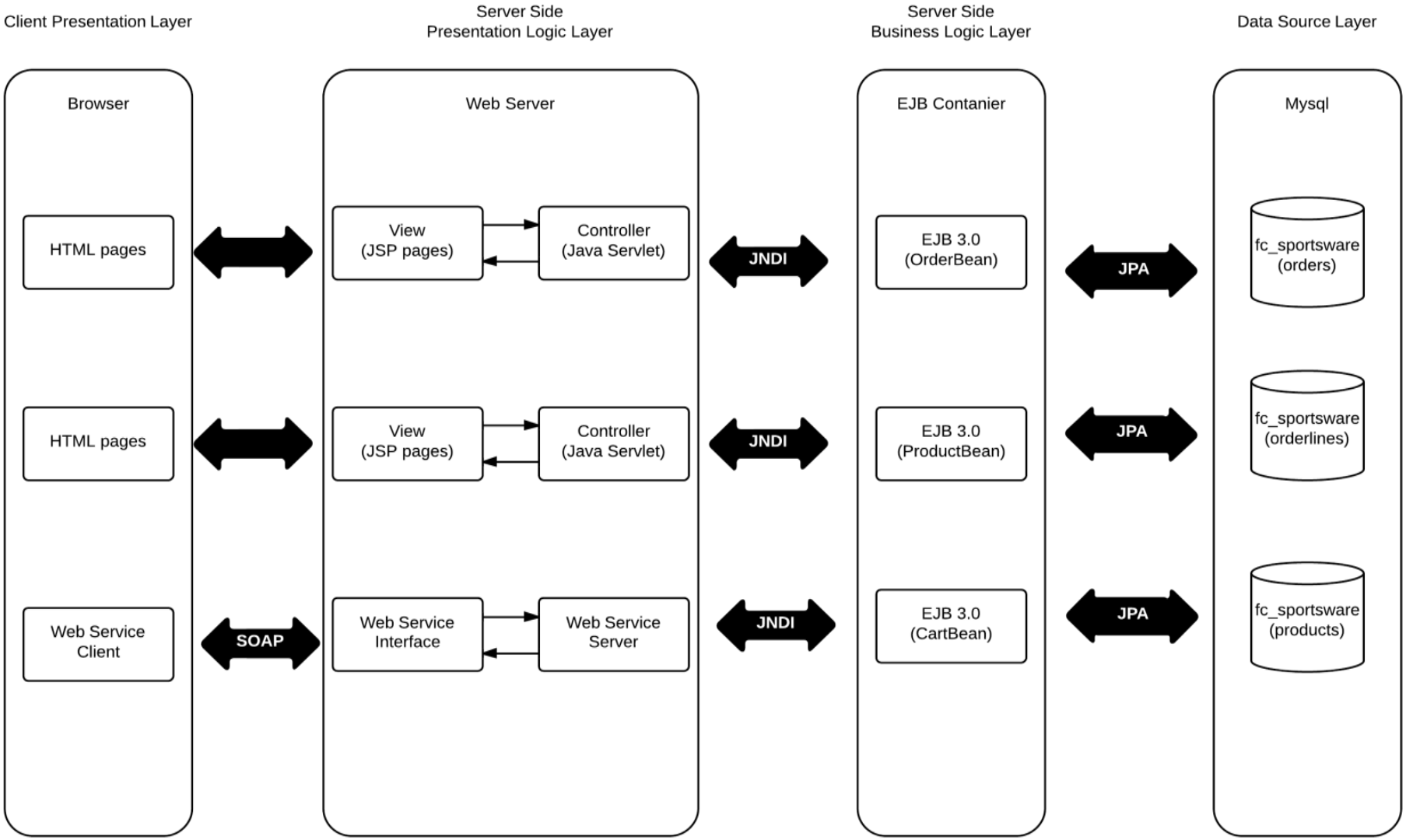
EJB: The current version that we are using for this application is EJB3.0 which enable us to accomplish all the functional and non-functional requirements like security and scalability. Using EJB module also split the project into presentation layer and business logical layer, it makes the application with higher security and easy to maintain.

Web Application: For the web application project, the MVC architecture is implemented in the project. It ensures efficiency and consistency of the web application especially separating the project into different components allows us to design, implement and test it independently, but easy to organise it into one project.

Web Service: As a distributed computing platform, it enables other application or server to communicate with us easily over the network via SOAP through HTTP protocol. We have also added BASIC web-authentication to secure our web services to ensure that only desired user could access our web services.

Student Name: Guangbo Chen
Student ID: 11376860

System Architecture Diagram



Project Components description

JPA2.0 (Java Persistent API)

In this project we would use Hibernate 4.2.5 Final version as our JPA provider, and in order to use Hibernate under the Weblogic 10.3.x version, there are few configurations that we have to setup for its running environment and class path.

Firstly, we need to download the Hibernate 4.2.5 Final version from JBoss Community website and add its core library to the project class path (alternatively download it from Maven repository). Also install JPA2.0 version for the project when creating the Weblogic server.

Then adding the following configuration to the weblogic-application.xml file within the Enterprise Application project

```
<wls:prefer-application-packages>
        <wls:package-name>antlr.*</wls:package-name>
        </wls:prefer-application-packages>
</wls:weblogic-application>
```

Also we need to tell the Weblogic server to use Hibernate as persistence provider via adding few configurations to the persistence.xml file like below

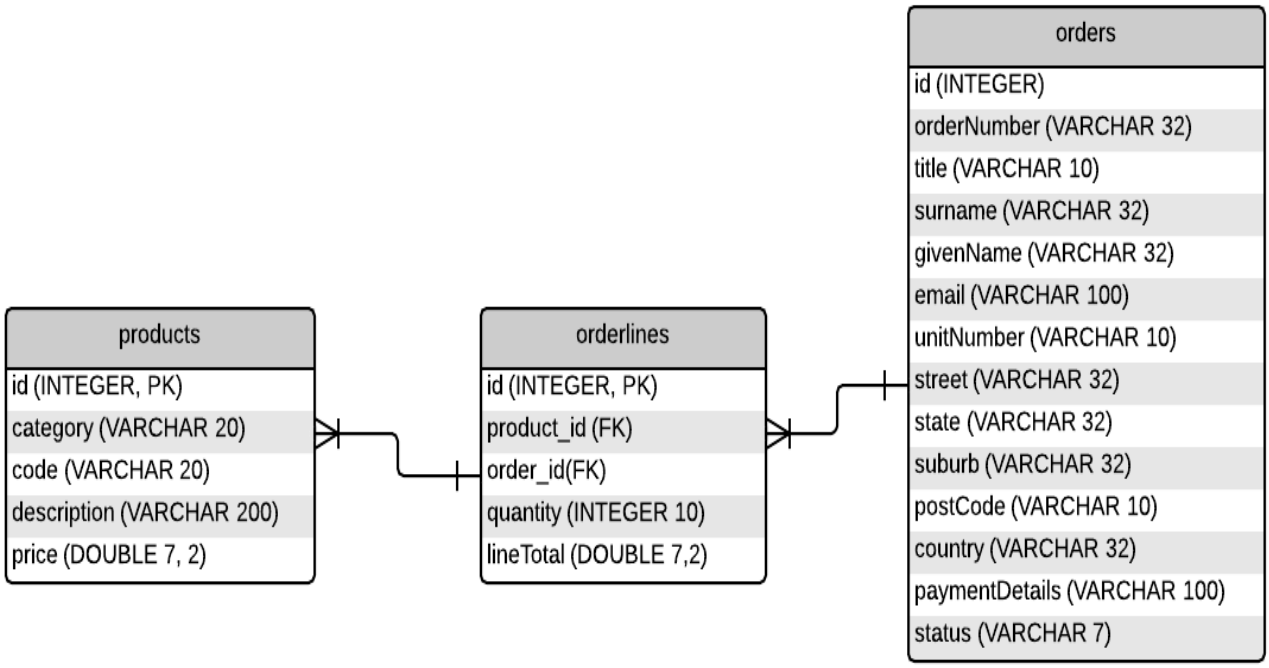
```
<persistence-unit name="fc_db" transaction-type="JTA">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
        <jta-data-source>jdbc/fcSportswareDB</jta-data-source>
        <properties>
                <property name="hibernate.transaction.jta.platform"
                value="org.hibernate.service.jta.platform.internal.WeblogicJtaPlatform"/>
        </properties>
</persistence-unit>
```

Lastly, in order to run JPA2.0 within Weblogic 10.3.x version we have to add two jar files to the Weblogic class path by modify the commEnv.sh(for Linux and Mac) or commEnv.cmd (for windows) with following path at the beginning (note this is not required if we are using Weblogic 12c)

```
PRE_CLASSPATH=”$MW_HOME/modules/javax.persistence_1.1.0.0_2.0.jar;$MW_HOME/modules/com.oracle.
jpa2support_1.0.0.0_2-1.jar”
```

MySQL Database (Data Source Layer)

Fc_sportsware Database scheme is located in the fc_web / WebContent / WEB-INF / schema.sql

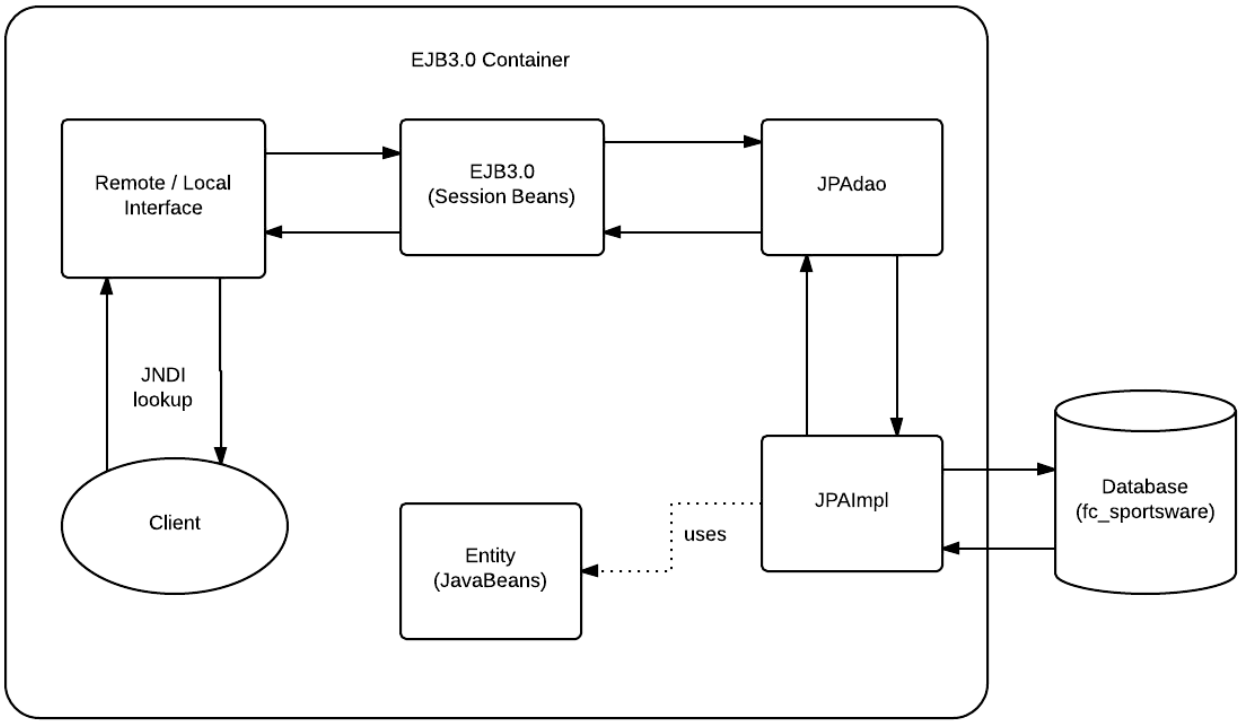


Each database table has mapped as an Entity in the Enterprise Web Application for implementing object-relational design. The Entity relationships between those tables are described as below:

Products with Orderlines: unidirectional one-to-one mapping, an orderlines would have maximum and minimum products, and so far the products doesn't requires to manage or connect to the orderlines, so no entity mapping is required from products to orderlines.

Orders and Orderlines: bidirectional on- to-many relationship, each order would contains many orderlines and each orderlines would only belong to one order.

EJB3.0 (server-side business logic)

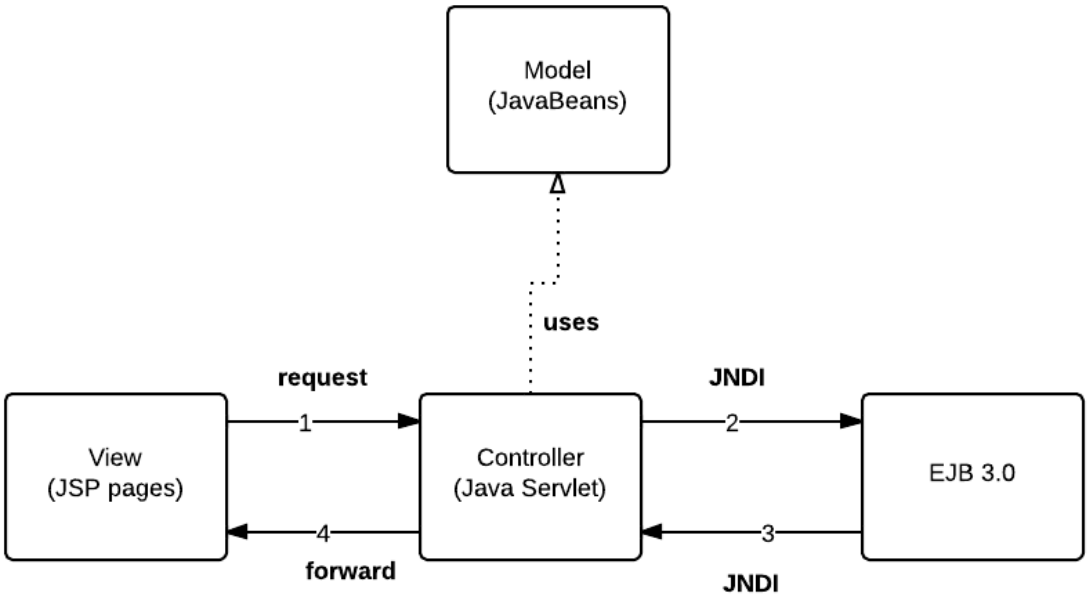


EJB project is presented as business logic layer in this project, it provides lifecycle operations like create remove and find, and its remote and local interface provides facets for client to utilise all EJB-based functionality.

Additionally, in this layer we going to run JPA within the EJB container via JAVA EE dependency injection (@PersistenceContext) and the Weblogic server will automatically doing the InitialContext and JDNI lookup for us, and the only thing we need to setup is add a Container-Entitymanager to the Weblogic server.

For those JPA Implementation classes it will manages the Entity lifecycle and its state via EntityManager API operation like persist, merge, remove and create.

Web Application (server-side presentation logic)

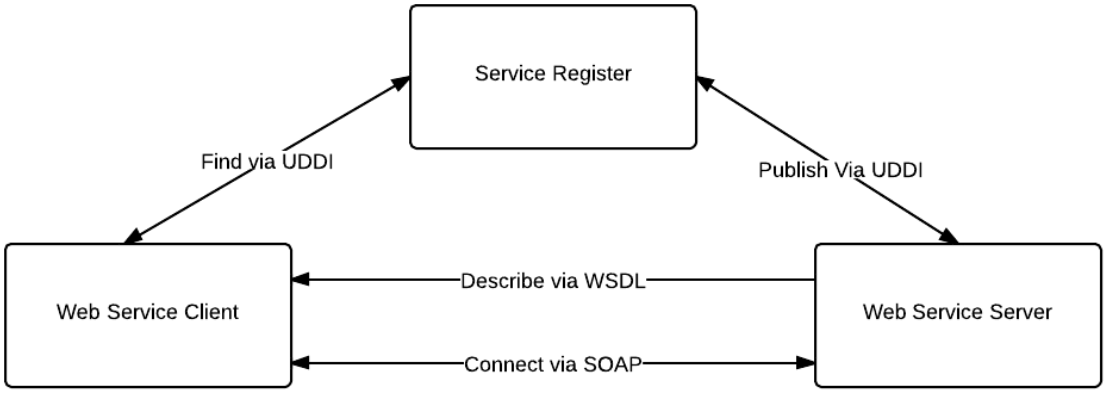


The fc_web project is presented as server-side presentation logic layer within the fc_sportsware Enterprise Application Project, the main feature for this layer is to handle the http request from the view layer (normally regarded as JSP pages), and it retrieves the data via communicating with EJB container through JNDI lookup, finally it forwards the request to the client page.

Additionally, in this layer our project would use the MVC framework to enhance the class encapsulation and polymorphism, as it would also make our code highly reusable and maintainable, also because in this layer we are calling the EJB session beans (both stateless and statefull), so we have to manage the EJB dependency injection.

Lastly for the view layer, our project would use the bootstrap3 CSS template and JSTL for facilitating the interface design.

Web Services (Supplier Web Service Server)



The supplier web service server that we have built for this application allows other application or server to communicate with us via SOAP through HTTP protocols. The data is exchanged in XML format, it also required BASIC authentication when user intends to communicate with the server, so in order to connect to the server, user has to include their username and password in their SOAP message header while the server would validate it by the information they have send.

Transaction

Web Transaction: not required

No transaction is required in our web server side, as our web server is only for formatting and forwarding the data, it does not carrying out any business logic, therefore any programmatic web transaction should be implemented.

EJB Transaction:

In our application we would use Declarative transaction for our EJB module

Stateful Shopping Cart Bean: using JPA EntityManager with "Extended Persistence Context", it ensures that we would have the persistent ACID properties over a shopping cart. Also for the EJB Transaction attributes we would use "Requires_new" property to ensure its will initiates its own transaction scope regardless of being called within existing transaction.

Stateless Product EJB Bean: the class level transaction is described as "supports" which would joins the transaction scope if invoke as part of transaction, the reason for this is because this EJB Bean in only implement with getting data action, therefore leave the container to manages it would be the best choice to avoid overhead.

Stateless Order EJB Bean: the class level transaction attributes is described as "required" for safe bet, for some method level with action of updating or changing database the transaction attributes is

described as “Requires_new” as the method is invoked from transactional context, and we want to stress its correctness and data integrity.